Automatic Verification of Security Protocols Using Approximations

Yohan Boichut, Pierre-Cyrille Héam, Olga Kouchnarenko

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Automatic Verification of Security Protocols Using Approximations

Yohan Boichut — Pierre-Cyrille Héam — Olga Kouchnarenko

N° 5727

Octobre 2005

THÈME 2

ISSN 0249-6399 ISRN INRIA/RR--5727--FR+ENG

Automatic Verification of Security Protocols Using Approximations

Yohan Boichut* , Pierre-Cyrille Héam** , Olga Kouchnarenko ***

Thème 2 — Génie logiciel
et calcul symbolique
Projet CASSIS

Rapport de recherche n° 5727 — Octobre 2005 — 27 pages

Abstract: Security protocols are widely used in open modern networks to ensure safe communications. It is now recognized that formal analysis can provide the level of assurance required by both developers and users of the protocols. Unfortunately it is generally undecidable to certify whether a protocol is safe or not. However the automatic verification of security protocols can be attempted using abstraction-based approximation. For this purpose, tree automata approximations were introduced by Genet and Klay in 2000. In this paper, we propose an extension of their techniques making the approach efficiently automatic. Our contribution has been implementing in the `TA4SP` tool with a high level specification language as input format, providing positive practical results on industrial security protocols.

Key-words: Security protocols, verifications, approximations, abstractions

* LIFC – Université de Franche-Comté, email: boichut@univ-fcomte.fr
** LIFC – Université de Franche-Comté, email: heampc@univ-fcomte.fr
*** LIFC – Université de Franche-Comté, email: kouchna@univ-fcomte.fr

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy (France)
Téléphone : +33 3 83 59 30 00 —Télécopie : +33 3 83 27 83 19

Approximations pour la vérification automatique de protocoles de sécurité

Résumé : De nos jours, les protocoles de sécurité (ssh, https,...) sont intensivement utilisés. Nous nous intéressons à la vérification de tels protocoles d'un point de vue symbolique : même si l'on suppose que les primitives cryptographiques utilisées sont parfaites, des failles peuvent néanmoins intervenir à cause de la parallelisation des sessions. C'est ce qu'on appelle par exemple les "man in the middle attacks". Nous avons pour objectif de vérifier qu'un protocole ne comporte pas de telles failles.

D'un point de vue général, même dans des cas très restreints, ce problème est indécidable pour un nombre non borné de sessions. Nous utilisons alors une méthode introduite par Genet et Klay utilisant des approximations et des abstractions. Nous montrerons comment cette méthode peut être automatisée pour devenir semi-algorithmique et pour fournir, en pratique, des résultats concluants. Nous présenterons aussi l'outil TA4SP développé dans ce cadre et utilisant un langage de spécification (HLPSL) de haut niveau.

Mots-clés : Protocoles de sécurité, vérification, approximations, abstractions

1 Introduction

1.1 Context

It is now widely accepted that security protocols are complex objects, which should be specified, designed and verified in a formal way. Many theoretical works have been devoted to this problem, and useful results have been obtained. In order to verify security protocols in an automatic way, this paper provides a new theoretical framework inspired by the abstraction-based approximation method developed by Genet and Klay in [12]. A tool named `TA4SP`[1], implementing this framework, and practical results on industrial security protocols are also presented.

Formal verification techniques for security protocols can be roughly classified in three categories. The first category relies on an exploration of the reachable state space using general model-checking techniques [13,6,14]. However, they only allow the verification of finite state systems. In order to apply these techniques, the behavior of the intruder is arbitrarily bounded. Consequently, these verification methods are not sound, but can find bugs, though.

The second (and well-known) category for the verification of security protocols relies on the theorem-proving techniques. Among them, we find for instance Paulson's work [17] with the `ISABELLE` prover, and Bolignano's work [3] with `Coq`. With these techniques, the protocol correctness is established for general cases; for example with an arbitrary number of parallel sessions. Unfortunately, this approach needs large human expertise even for simple protocols.

The third category is based on the constraint solving problem techniques. Indeed, rewriting systems and constraint solving are adequate tools for the proof of security properties protocols [10,20,5]. This approach is as efficient as the finite model-checking one.

Current verification techniques based on model-checking method can be used whenever the number of agents and the number of sessions between the agents (and hence the state-space) are finite. In this case, the protocol security problem is co-NP-complete [18].

However, in a practical use, one cannot bound the number of sessions. So, it would be desirable to verify protocols without these restrictions. But in this case, the problem becomes undecidable. Abstractions [9,12] represents an attractive way to tackle this problem. Abstraction techniques are intensively used in program analysis and the verification of reactive and hardware systems. The use of abstractions often provides a way to prove the correctness/security of systems by over-estimating the possibility of failure. Moreover, since security protocols are of industrial interest, it is relevant to develop fully automatic techniques and tools accepting an intuitive input format. In this paper, we propose a formal approach to automatically verify security protocols in an unbounded sessions number context and we present its implementation: `TA4SP`.

[1] Tree Automata based Automatic Approximations for the Analysis of Security Protocols developed in LIFC.

1.2 Contributions

This paper describes extensions to Genet and Klay's method for security protocols based on tree automata. The motivation of this paper is to obtain an efficient automatic tool dedicated to the verification of security protocols. The challenge is in the combination of the three points: effectiveness (fast computation of the over-approximation in practical cases), automaticity and precision (the semi-algorithmic approach used has to be practically relevant).

Our contributions are as follows.

- Generally, security protocols cannot be encoded by left-linear term rewriting systems. The left-linearity condition like in [11] is convenient to protocols however it has to be verified on the fix point automaton (denoted $\mathcal{A}_k$ in Section 2.1). If it does not satisfy it, the method is not safe, i.e. the automaton $\mathcal{A}_k$ does not recognize an over-approximation of accessible terms. We know *a posteriori* whether the computation is pertinent or not. In this paper, we define a new left-linearity-like condition based on constant typing, which is easily checked only on the initial automaton (*a priori*) and expressive enough to handle security protocols.
- The method described in [12,11] is not automatic and has to be applied with the help of an expert. Contrary to this method, we define an automatic way to generate an approximation function which provides positive results on several industrial security protocols in a reasonable time (less than few minutes on a usual computer).
- Since security protocols are widely used by industrials, we are interested in developing a tool that can be used by a non expert. So, we have been implementing a connection of `TA4SP` to a high level and intuitive specification language like `HLPSL` [4].

1.3 Layout of the Paper

This paper is organized as follows. A quick introduction to the Genet and Klay method [12] and basic definitions are given in Section **??**. Sections 3 and 4 respectively describe the notion of *symbolic normalisation* and a new left linearity criterion well-adapted to the specification of security protocols (a related completion algorithm is also given). Thus, we show how to automatically generate approximations functions in Section 5. Finally, we present in Section 6 the `TA4SP`[2] tool (implementing the approach) and practical experimentations obtained.

2 Preliminaries

2.1 Related Work and Genet and Klay Approach

The main idea of Genet and Klay [12] is to build an automaton whose language represents an over-approximation of the network's configuration with an unbounded number of sessions.

[2] Tree Automata based Automatic Approximations for the Analysis of Security Protocols developed in LIFC.

The automaton $\mathcal{A}_0$ represents the initial configuration of the network and different actions the intruder is able to do (composition of messages and encryption). A TRS $\mathcal{R}$ defines the abilities of the intruder to analyze messages, and each step of the protocol ($l \rightarrow r$: l is the message received, and r the message sent). A term rewriting system $\mathcal{R}$ is *runnable* if for each rule $l \rightarrow r \in \mathcal{R}$, $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l)$ ($\mathcal{V}\mathrm{ar}(t)$ is the set of variables appearing in the term t), i.e., if the TRS is left-linear (a weaker left-linear-like condition is proposed in [11]). The Timbuk[3] tool computes, for a given initial automaton, a given TRS and a given approximation function γ, an automaton $\mathcal{A}_k$. To summarize the algorithm, starting from the initial automaton, Timbuk finds a rule $l \rightarrow r \in \mathcal{R}$ and a substitution σ such that $l\sigma \in \mathcal{L}(\mathcal{A}_i)$ and $r\sigma \notin \mathcal{L}(\mathcal{A}_i)$. Notice that one requires $\mathcal{V}\mathrm{ar}(r) \subseteq \mathcal{V}\mathrm{ar}(l)$ to ensure $r\sigma$ to be a ground term. Then, the approximation function γ helps to normalize $r\sigma$ by adding needed transitions to the current automaton. Finally, $\mathcal{A}_{i+1}$ recognizes $r\sigma$ and the computation goes on, until reaching a fix point $\mathcal{A}_k$ if it exists.

The language recognized by $\mathcal{A}_k$ represents an over-approximation of the set of messages that the intruder may know. To check whether the intruder may know a secret, we represent the set of secret terms (i.e., the terms we don't want the intruder to be able to access) by a tree automaton $\mathcal{A}_{\mathrm{secret}}$. We compute the intersection between $\mathcal{L}(\mathcal{A}_{\mathrm{secret}})$ and $\mathcal{L}(\mathcal{A}_k)$ for classical methods on tree automata. If the intersection is empty then the protocol is safe (we are sure the intruder has no access to secret terms), otherwise one cannot conclude. See Fig. 1 for an illustration.

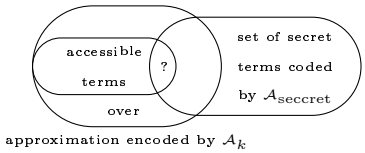

Fig. 1. Approximation of accessible terms

Recently, the above approach was extended to AC-tree automata (tree automata using associative and commutative symbols) and implemented in the `ACTAS` tool [16].

Our improvements allow the method described in [12] to become fully automatic thank to automatically generated approximations. This generation is based on a notion of symbolic normalisation presented in the following section.

2.2 Background and Notations

For a definition of tree automata, the reader may refer to [7]. A tree automaton is finite if its set of transitions is finite.

[3] An `OCAML` tree automata library developed by T. Genet at IRISA-Rennes under GPL.

For any set A, we denote by 2^A the set of subsets of A.

Let $\mathcal{F}$ be a finite set of symbols with their arities. The set of symbols of $\mathcal{F}$ with arity i is denoted $\mathcal{F}_i$. Let $\mathcal{X}$ be a finite set whose elements are called variables. We assume that $\mathcal{X} \cap \mathcal{F} = \emptyset$.

We denote by $\mathbb{N}$ the set of natural integers and $\mathbb{N}^*$ denotes the set of finite strings over $\mathbb{N}$.

A finite ordered tree t over a set of labels $(\mathcal{F}, \mathcal{X})$ is a function from a prefix-closed set $\mathcal{P}os(t) \subseteq \mathbb{N}^*$ to $\mathcal{F} \cup \mathcal{X}$. A term t over $\mathcal{F} \cup \mathcal{X}$ is a labeled tree whose domain $\mathcal{P}os(t)$ satisfies the following properties:

- $\mathcal{P}os(t)$ is non-empty and prefix closed;
- For each $p \in \mathcal{P}os(t)$, if $t(p) \in \mathcal{F}_n$ then $\{i \mid p.i \in \mathcal{P}os(t)\} = \{1, \ldots, n\}$;
- For each $p \in \mathcal{P}os(t)$, if $t(p) \in \mathcal{X}$ then $\{i \mid p.i \in \mathcal{P}os(t)\} = \emptyset$.

Each element of $\mathcal{P}os(t)$ is called a position of t. A frontier position is a position p such that for all $\alpha \in \mathbb{N}$, $p.\alpha \notin \mathcal{P}os(t)$. The set of frontier positions of t is denoted $\mathcal{FP}os(t)$. Each position p of t such that $t(p) \in \mathcal{F}$ is called a functional position and is denoted $\mathcal{P}os_{\mathcal{F}}(t)$. The set of terms over $(\mathcal{F}, \mathcal{X})$ is denoted $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term t is ground if t does not contain any variables.The set of ground terms is denoted $\mathcal{T}(\mathcal{F})$.

A subterm $t_{|p}$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ at position p is defined by:

- $\mathcal{P}os(t_{|p}) = \{i \mid p.i \in \mathcal{P}os(t)\}$, and
- For all $j \in \mathcal{P}os(t_{|p})$, $t_{|p}(j) = t(p.j)$.

We denote by $t[s]_p$ the term obtained by replacing in t the subterm $t_{|p}$ by s.

If $\mathcal{X}$ contains n elements and is (arbitrarily) ordered then a context C is an element of $\mathcal{T}(\mathcal{F}, \mathcal{X})$ in which all elements of $\mathcal{X}$ occur exactly once. The expression $C[t_1, \ldots, t_n]$ for $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ denotes the term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ obtained from C by replacing the i-th element of $\mathcal{X}$ by t_i for each $1 \leq i \leq n$.

For every sets A and B, we denote by $\Sigma(A, B)$ the set of functions from A to B.

If $\sigma \in \Sigma(\mathcal{X}, B)$ and $B \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ then for each term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we denote by $t\sigma$ the term obtained from t by replacing the variable x by $\sigma(x)$, for each $x \in \mathcal{X}$.

A term rewriting system (TRS) over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is a finite set of pair (l, r) of $\mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X})$, denoted $l \to r$, such that the set of variables occurring in r is included in the set of variables of l. A TRS is left-linear [resp. right-linear] if in each rule $l \to r$, all variables occurring in l [resp. r] occur at most once in l [resp. r].

In this paper, the expression *left-linear-like condition* is used to denote a condition on TRS that is intuitively closed to the notion of *left-linearity*.

In this paper some elements are fixed:

- $\mathcal{R}$ is a TRS over $\mathcal{T}(\mathcal{F}, \mathcal{X})$.
- $\mathcal{Y} = \{y_{l \to r,p} \mid l \to r \in \mathcal{R},\ p \in \mathcal{P}os_{\mathcal{F}}(r)\}$ is a finite set of variables. We assume that $\mathcal{X} \cap \mathcal{Y} = \emptyset$ and $\mathcal{F} \cap \mathcal{Y} = \emptyset$.
- $\mathcal{Q}$ denotes an infinite ordered set whose elements are called *states*.

3 Symbolic Normalisation

Informally, the goal of the normalisation is the following: if in a finite tree automaton $\mathcal{A}$ one has $l\sigma \rightarrow_{\mathcal{A}}^{*} q$ and $r\sigma \not\rightarrow_{\mathcal{A}}^{*} q$, we add to the set of transitions of $\mathcal{A}$ a finite set of transitions N, called the normalisation set of the symbolic transitions $r\sigma \rightarrow q$ such that $r\sigma \rightarrow_{N}^{*} q$. From a practical point of view, the ways to normalize two symbolic transitions $r\sigma_1 \rightarrow q_1$ and $r\sigma_2 \rightarrow q_2$ (related to the same rule $l \rightarrow r$) are very similar, leading to the symbolic normalisation notion.

Definition 1. *A symbolic transition (t, y) is an element of the set $\mathcal{T}(\mathcal{F}, \mathcal{X} \cup \mathcal{Y}) \times (\mathcal{Y} \cup \mathcal{Q})$, written $t \rightarrow y$, such that $t = f(y_1, \ldots, y_n)$ with $f \in \mathcal{F}$, and $y_i \in \mathcal{X} \cup \mathcal{Y}$ for each $1 \leq i \leq n$.*

For example, $g(x, y) \rightarrow q$ is a symbolic transition where $g \in \mathcal{F}$ and $x, y \in \mathcal{X}$. We consider a symbolic state as a variable in $\mathcal{Y}$.

Thus, we propose a symbolic normalisation allowing a reduction of a term in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ corresponding to the right hand side r of a rule $l \rightarrow r$ of $\mathcal{R}$, either to a state or to a symbolic state.

Definition 2. *Given $l \rightarrow r \in \mathcal{R}$ and $q \in \mathcal{Q}$, the symbolic normalisation of the transition $r \rightarrow q$ modulo $l \rightarrow r$ is the set of symbolic transitions, written* $\mathrm{Norm}(r \rightarrow q, l \rightarrow r)$*, defined by*

$$
\begin{aligned}
\mathrm{Norm}(r \rightarrow q, l \rightarrow r) \; = \{ & r(p)(z_{p.1}, \ldots, z_{p.n}) \rightarrow \alpha_p \mid \\
& p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(r), \\
& r(p) \in \mathcal{F}, \\
& z_{p.i} = \begin{cases} r(p.i) & \text{if } r(p.i) \in \mathcal{X} \wedge \\ & \quad p.i \in \mathcal{F}\mathcal{P}\mathrm{os}(r) \\ y_{l \rightarrow r, p.i} & \text{otherwise,} \end{cases} \\
& \alpha_p = \begin{cases} y_{l \rightarrow r, p} \in \mathcal{Y} & \text{if } p \neq \epsilon, \\ q & \text{otherwise} \end{cases} \\
& \}.
\end{aligned}
$$

Notice that the variable $y_{l \rightarrow r, p.i}$ represents the symbolic state associated to the rule $l \rightarrow r$ and the position $p.i$. The following example illustrates Definition 2 assuming $r = g(x, g(f(a), f(z))$ where $a, g, f \in \mathcal{F}$ and $x, z \in \mathcal{X}$.

$$
\begin{aligned}
\mathrm{Norm}(r \rightarrow q, l \rightarrow r) = \{ & f(z) \rightarrow y_{l \rightarrow r, 2.2}, \\
& a \rightarrow y_{l \rightarrow r, 2.1.1}, \\
& f(y_{l \rightarrow r, 2.1.1}) \rightarrow y_{l \rightarrow r, 2.1}, \\
& g(y_{l \rightarrow r, 2.1}, y_{l \rightarrow r, 2.2}) \rightarrow y_{l \rightarrow r, 2}, \\
& g(x, y_{l \rightarrow r, 2}) \rightarrow q \}.
\end{aligned}
$$

For instance, one has $f(z) \rightarrow y_{l\rightarrow r,2.2}$ since the symbol f – corresponding to the subterm $f(z)$ – is at position 2.2 in r.

Let E be a set of symbolic transitions. We denote by $\mathcal{V}\mathrm{ar}(E)$ the set of variables occurring in the elements of E, i.e., $\mathcal{V}\mathrm{ar}(E) = \{y \in \mathcal{Y} \mid \exists t \in \mathcal{T}(\mathcal{F}, \mathcal{Y}),\ t \rightarrow y \in E\}$. We denote by $E\sigma$ the set $\{t\sigma \rightarrow y\sigma \mid t \rightarrow y \in E\}$ of transitions.

Definition 3. *An approximation function γ is a function $\gamma : \mathcal{R}\times\Sigma(\mathcal{Q},\mathcal{X})\times\mathcal{Q} \mapsto \{\Sigma(\mathcal{Q},\mathcal{W}) \mid \mathcal{W} \subseteq \mathcal{Y}\}$ such that $\gamma(l \rightarrow r, \sigma, q) \in \Sigma(\mathcal{Q}, \mathcal{V}\mathrm{ar}(\mathrm{Norm}(r \rightarrow q, l \rightarrow r)))$.*

For example, consider $r = g(x, g(f(a), f(z))$. For each σ and each q, $\gamma(l \rightarrow r, \sigma, q)$ is a function from $\{x, z, y_{l\rightarrow r,2.1.1}, y_{l\rightarrow r,2.1}, y_{l\rightarrow r,2.2}, y_{l\rightarrow r,2}\}$ to $\mathcal{Q}$.

Definition 4. *Let γ be an approximation function, Δ a set of transitions, $l \rightarrow r \in \mathcal{R}$ and $\sigma \in \Sigma(\mathcal{X}, \mathcal{Q})$ such that $l\sigma \rightarrow^*_\Delta q$. The γ-normalisation of the symbolic transition $r\sigma \rightarrow q$, written $\mathrm{Norm}_\gamma(r\sigma \rightarrow q, l \rightarrow r)$, is defined by*

$$\mathrm{Norm}_\gamma(r\sigma \rightarrow q, l \rightarrow r) = [\mathrm{Norm}(r \rightarrow q, l \rightarrow r)\sigma]\, \gamma(l \rightarrow r, \sigma, q).$$

For example, let us consider $r = g(x, g(f(a), f(z))$ with $\sigma(x) = q_1$, $\sigma(z) = q_2$, and $\gamma = \gamma(l \rightarrow r, \sigma, q)$. Assume that $\gamma(y_{l\rightarrow r,2.1.1}) = q_1$, $\gamma(y_{l\rightarrow r,2.1}) = q_3$, $\gamma(y_{l\rightarrow r,2.2}) = q$, $\gamma(y_{l\rightarrow r,2}) = q_1$. Then

$$\begin{aligned}\mathrm{Norm}_\gamma(r\sigma \rightarrow q, l \rightarrow r) = \{&f(q_2) \rightarrow q,\\ &a \rightarrow q_1,\\ &f(q_1) \rightarrow q_3,\\ &g(q_3, q) \rightarrow q_1,\\ &g(q_1, q_1) \rightarrow q\}.\end{aligned}$$

Lemma 1. *Let γ be an approximation function, Δ a set of transitions, $l \rightarrow r \in \mathcal{R}$ and $\sigma \in \Sigma(\mathcal{X}, \mathcal{Q})$ such that $l\sigma \rightarrow^*_\Delta q$. One has*

$$r\sigma \rightarrow^*_{\Delta\cup\mathrm{Norm}_\gamma(r\sigma\rightarrow q, l\rightarrow r)} q.$$

Proof. Obvious by Definition 2 and because a substitution covers the symbolic state set corresponding to the rule $l \rightarrow r$.

4 Completion Algorithm and Left-Linearity

By definition, a term t can be rewritten into a term s by a rule $l \rightarrow r$ if there exist a position p of t and a substitution μ of the variables of t by *ground terms* such that $t = t_p[l\mu]$ and $s = t_p[r\mu]$. The idea of the normalisation is to substitute variables by *states* in order to compute accessible terms from regular tree language by a TRS. Using state substitutions

rather than term substitutions is correct if at each step of the computation the automaton $\mathcal{A}_i$ (with the notation of Section 2.1) and $\mathcal{R}$ satisfy the following condition: *For each rule of* $\mathcal{R}$, $l \to r$, *each state* q *of* $\mathcal{A}_i$, *and each substitution* μ *of the variables of* l *by terms, if*

$$l\mu \to^*_{\mathcal{A}_i} q$$

then there exists a substitution σ *of the variables of* l *by states such that*

$$l\mu \to^*_{\mathcal{A}_i} l\sigma \to^*_{\mathcal{A}_i} q.$$

In general, the above property is not easy to check (and sometimes false) for TRS. If the TRS is left-linear, this condition is trivially satisfied (for more details, see [11]). However, modeling security protocols by TRS requires some non-left-linear rules. To solve this problem, Genet et al. in [11], propose another left-linear-like condition (implying the above condition) that has to be (not trivially) checked either at each step of the completion or on the final fix point automaton. We want to emphasize the fact that these conditions are not well-adapted for practical computations: verifying the condition requires time and if the condition is not satisfied, the computation has to be done again with another approximation function.

We define in this section a new left-linear-like condition that guarantees the correctness of the computation. This condition is restrictive on the TRS, but

1. is expressive enough for many industrial security protocols,
2. has to be checked only on the initial automaton,
3. is easy to check.

In practice (for security protocols modeling), non left-linear variables often concern either the identity of the agents or numbers for nonces/keys, which are constants in the framework of nonce abstraction we have been implementing. Informally, we introduce a kind of typing on constants, variables and states to ensure the validity of the computation.

The rest of this section is organized as follows.

The first part is dedicated to the formal definition of this typing (Definition 5) and its technical properties (Lemmas 2 and 3). Then, we define the completion procedure in Definition 6. Finally, the correctness of this procedure is proved in Theorem 1 and Proposition 1.

Definition 5. *Let* $\mathcal{Z} \subseteq \mathcal{X}$, $F \subseteq \mathcal{F}_0$, *and* $f \in \mathcal{F}_1$. *Let* $\mathcal{A}$ *be a finite tree automaton, and* γ *be an approximation function. We say that the pair* $(\mathcal{A}, \gamma)$ *is* $(F, \mathcal{Z}, f)-$*compatible if the following conditions are satisfied:*

1) $\forall A \in F$, *there exists a unique state of* $\mathcal{Q}$, *denoted* q_A, *such that* $A \to_\Delta q_A$, *the function* $F \mapsto \mathcal{Q}$, *where* $A \mapsto q_A$, *is injective, and* $\mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A}) = \{q_A \mid A \in F\}$.
2) $\forall t \to q \in \Delta$, *if* $t = h(q_1, \ldots, q_n)$ *and* $\exists i \in [1, \ldots, n]$ *such that* $q_i \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$, *then* $n = i = 1$ *and* $h = f$.
3) $\forall t \to q \in \Delta$, *if* $q \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ *then* $t \in F$.
4) $\forall t \to q \in \Delta$, *if* $t = f(q')$ *then* $q' \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$.
5) $\forall \mathrm{l} \to \mathrm{r} \in \mathcal{R}$, $\forall p \in \mathcal{P}\mathrm{os}(l)$, $l(p) \in \mathcal{Z} \cup F \iff \exists p' \in \mathcal{P}\mathrm{os}(l)$ *such that* $p = p'.1$ *and* $l(p') = f$.

6) $\forall \mathrm{l} \to \mathrm{r} \in \mathcal{R}, \forall p \in \mathcal{P}\mathrm{os}(r)$, $r(p) \in \mathcal{Z} \cup F \iff \exists p' \in \mathcal{P}\mathrm{os}(r)$ *such that* $p = p'.1$ *and* $r(p') = f$.

7) $\forall l \to r \in \mathcal{R}, \forall p \in \mathcal{P}\mathrm{os}_{\mathcal{F}}(r)$, $\forall \sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$, $\forall q \in \mathcal{Q}, r(p) \in F \iff \gamma(l \to r, \sigma, q)(y_{l \to r, p}) = q_{r(p)} \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$.

Let us informally explain the above definition. Point *1)* ensures that every typed variable is one-to-one associated to a state of the automaton. Points *2-3-4)* ensure that this association is compatible with the transitions of the automaton: states defined in point *1)* can only be used to reduce typed constants. Points *5-6-7)* ensure that the previous points are preserved when applying the completion procedure (defined in Definition 6).

The two following lemmas are useful to prove that one can use state substitution rather than term substitution by only verifying a condition on the initial automaton (Theorem 1).

Lemma 2. *Let* $(\mathcal{A}, \gamma)$ *be a* $(F, \mathcal{Z}, f)$*-compatible pair. Let* $l \to r \in \mathcal{R}$, $q \in \mathcal{Q}$ *and* $\mu \in \Sigma(\mathcal{T}(\mathcal{F}), \mathcal{X})$ *such that* $l\mu \to^*_{\mathcal{A}} q$. *If there exists a frontier position* p *of* l *such that* $l(p) \in \mathcal{Z}$, *then* $\mu(l(p)) \in F$.

Proof. Since $l(p) \in \mathcal{Z}$ and since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible (by condition *5* of Definition 5), there exists a position p' of l such that $p = p'.1$ and $l(p') = f$. Consequently $l_{|p'} = f(l_{|p})$. Furthermore $l\mu \to^*_{\mathcal{A}} q$. Thus there exist $q_1, q_2 \in \mathcal{Q}$ such that

$$\mu(l(p)) \to^*_{\mathcal{A}} q_1 \quad \textit{and} \quad f(q_1) = q_2.$$

Since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible (by conditions *4* and *3* of Definition 5), one obtains $q_1 \in \mathcal{Q}_{\mathcal{F},\mathcal{Z}}$ and then $\mu(l(p)) \in F$.

Lemma 3. *Let* $(\mathcal{A}, \gamma)$ *be a* $(F, \mathcal{Z}, f)$*-compatible pair. Let* $l \to r \in \mathcal{R}$, $q \in \mathcal{Q}$ *and* $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ *such that* $l\sigma \to^*_{\Delta} q$. *One has* $q \notin \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ *and for all* $x \in \mathcal{V}\mathrm{ar}(l)$, *if* $\sigma(x) \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ *then* $x \in \mathcal{Z}$.

Proof.
– We first prove by contradiction that $q \notin \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. Assume that $q \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. Since $l\sigma \to^*_{\Delta} q$, there exist $q'_1, \ldots, q_n, ' \in \mathcal{Q}$ such that

$$l(\epsilon)(q'_1, \ldots, q'_n) \to_{\Delta} q.$$

One obtains $l(\epsilon) \in F$ because $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible and $q \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. A contradiction with Definition 5, condition *5*, and we are done.

– Let $x \in \mathcal{V}\mathrm{ar}(l)$ such that $\sigma(x) \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. Since $l\sigma \to^*_{\Delta} q$, there exist $q_1, \ldots, q_n, q' \in \mathcal{Q}$, $1 \leq k \leq n$, $p' \in \mathcal{P}\mathrm{os}(l)$ such that $l(p) = x$, $p = p'.1$, $q_k = \sigma(x)$ and

$$l(p')(q_1, \ldots, q_n) \to_{\Delta} q'.$$

Since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible, it implies $n = 1$ and $l(p') = f$. Consequently, $l(p) \in \mathcal{Z} \cup F$. Thus $x \in \mathcal{Z}$.

One can now define the completion procedure. Notice that it is very closed to the one described in [11], but well adapted to the definition of the symbolic normalisation.

Definition 6. *Let $\mathcal{A} = (\mathcal{Q}, \Delta, \mathcal{Q}_f)$ be a tree automaton and γ an approximation function. We denote $f_\gamma(\mathcal{A}) = (\mathcal{Q}', \Delta', \mathcal{Q}_f)$ the finite tree automaton defined by*

$$\Delta' = \Delta \cup \bigcup_{l \to r \in \mathcal{R}, \sigma \in \Sigma(\mathcal{Q},\mathcal{X}), q \in \mathcal{Q}, l\sigma \to^*_\Delta q, \neg r\sigma \to^*_\Delta q} \mathrm{Norm}_\gamma(r\sigma \to q, l \to r),$$

and

$$\mathcal{Q}' = \mathcal{Q} \cup \{q_i \mid \exists t \to q_i \in \mathrm{Norm}_\gamma(r\sigma \to q, l \to r) \wedge t \in \mathcal{T}(\mathcal{F})\mathcal{Q}\}.$$

To prove the main result of this section, we need the following lemma.

Lemma 4. *Let $\mathcal{Z} \subseteq \mathcal{X}$, $F \subseteq \mathcal{F}_0$, and $f \in \mathcal{F}_1$. Let $\mathcal{A}$ be a finite tree automaton and γ be an approximation function. If $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible then $(f_\gamma(\mathcal{A}), \gamma)$ is $(F, \mathcal{Z}, f)$-compatible too.*

Proof. We prove each condition of Definition 5.

1) Let $A \to q$ be a transition of $f_\gamma(\mathcal{A})$ such that $A \in F$. Since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible, if $A \to q \in \Delta$, then $q = q_A$. Assume that $A \to q \notin \Delta$. By Definition 6, there exist $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$, $q' \in \mathcal{Q}$, $l \to r \in R$ such that $A \to q \in \mathrm{Norm}_\gamma(r\sigma \to q, l \to r)$. Thus, by Definition 2, there exists $p \in \mathcal{P}\mathrm{os}_\mathcal{F}$ such that $r(p) = A$ and either $p = \epsilon$ and $q = q'$ or $p \neq \epsilon$ and $q = \gamma(l \to r, \sigma, q')(y_{l \to r, p})$. Since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible, $p \neq \epsilon$ by condition *6* of Definition 5. Consequently, $q = \gamma(l \to r, \sigma, q')(y_{l \to r, p})$. Therefore, $q = q_A$ by condition *7* of Definition 5, and we are done for condition *1*. Moreover, we obtain

$$\mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A}) = \mathcal{Q}_{F,\mathcal{Z},f}(f_\gamma(\mathcal{A})). \tag{1}$$

2) Let $h(q_1, \ldots, q_n) \to q$ be a transition of $f_\gamma(\mathcal{A})$. Assume that $q_\ell \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ with $1 \leq \ell \leq n$. Since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible and by Equality (1), if $t \to q \in \Delta$, then $n = \ell = 1$ and $h = f$. Otherwise, if $t \to q \notin \Delta$ then, by Definition 6, there exist $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$, $q' \in \mathcal{Q}$, $l \to r \in R$ such that $A \to q \in \mathrm{Norm}_\gamma(r\sigma \to q, l \to r)$. Thus, by Definition 2, there exists $p \in \mathcal{P}\mathrm{os}_\mathcal{F}$ such that $r(p) = h$ and there are two cases for q_ℓ.
   - $q_\ell = \sigma(r(p.\ell))$, then $r(p.\ell) \in \mathcal{Z}$ because $q_\ell \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ and by Lemma 3. Therefore, since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible by condition *6* of Definition 5, one has $r(p) = f$ and thus $n = \ell = 1$.
   - $q_\ell = \gamma(l \to r, \sigma, q)(y_{l \to r, p.\ell})$, then $q_\ell \in F$ because of $q_\ell \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ and since $(\mathcal{A}, \gamma)$ is $(F, \mathcal{Z}, f)$-compatible by condition *7* of Definition 5. Thus, using condition *7* of Definition 5 on $(\mathcal{A}, \gamma)$, one has $r(p) = f$ and $n = \ell = 1$.

   We are done for condition *2* of Definition 5.

3) Let $t \to q$ be a transition of $f_\gamma(\mathcal{A})$ such that $q \in \mathcal{Q}_{F,\mathcal{Z},f}(f_\gamma(\mathcal{A})) = \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. By Equality (1) and since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible, if $t \to q \in \Delta$ then $t \in F$. Assume that $t \to q \notin \Delta$. By Definition 6, there exist $\sigma \in \Sigma(\mathcal{Q},\mathcal{X})$, $q' \in \mathcal{Q}$, and $l \to r \in R$ such that $A \to q \in \mathrm{Norm}_\gamma(r\sigma \to q, l \to r)$. Thus, by Definition 2, there exists $p \in \mathcal{P}\mathrm{os}_\mathcal{F}$ such that $t = r(p)$ and either $t = r(\epsilon)$ and $q = q'$ or $q = \gamma(l \to r, \sigma, q)(y_{l\to r,p})$. If $t = r(\epsilon)$ then $l\sigma \to^*_\Delta q$. Therefore $q \notin \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ by Lemma 3, a contradiction. Consequently, $q = \gamma(l \to r, \sigma, q)(y_{l\to r,p})$. Since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible by condition *7* of Definition 5, one has $t \in F$.
4) Let $f(q') \to q$ be a transition of $f_\gamma(\mathcal{A})$. Since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible and by Equality (1), if $f(q') \to q \in \Delta$, then $q' \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$. Assume that $f(q') \to q \notin \Delta$. By Definition 6, there exist $\sigma \in \Sigma(\mathcal{Q},\mathcal{X})$, $q' \in \mathcal{Q}$, and $l \to r \in R$ such that $A \to q \in \mathrm{Norm}_\gamma(r\sigma \to q, l \to r)$. Thus, by Definition 2, there exists $p \in \mathcal{P}\mathrm{os}_\mathcal{F}$ such that $t = r(p)(q')$. Since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible by condition *6* of Definition 5, $r(p.1) \in \mathcal{Z} \cup F$.
   - If $r(p.1) \in \mathcal{Z}$ then $q' = \sigma(r(p.1)) \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ by Lemma 3.
   - If $r(p.1) \in F$ then $q' = q_{r(p.1)} \in \mathcal{Q}_{F,\mathcal{Z},f}(\mathcal{A})$ by part *1)* of the current proof.

5-6) Since conditions *5* and *6* of Definition 5 are conditions on $\mathcal{R}$ and γ, and since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible, $(f_\gamma(\mathcal{A}),\gamma)$ satisfies this condition too.
7) Since $(\mathcal{A},\gamma)$ is $(F,\mathcal{Z},f)$-compatible and by (1), $(f_\gamma(\mathcal{A}),\gamma)$ satisfies condition *7* of Definition 5.

Since the set of variables allowed to appear more than once in the left-hand side of a rule is known, one can easily define a linearity-like condition by Definition 7 as follows.

Definition 7. *Let $\mathcal{J} \subseteq \mathcal{X}$. A term rewriting system $\mathcal{R}$ is $\mathcal{J}-$left-linear, if for every rule $l \to r$ of $\mathcal{R}$, all $p,q \in \mathcal{FP}\mathrm{os}(l)$, $l(p) = l(q)$ and $l(p) \in \mathcal{J}$ imply $p = q$.*

Informally, a TRS is $\mathcal{J}-$left-linear if the variables of $\mathcal{J}$ may occur at most once in the left-hand sides of the rules.

Definition 8. *Let $\mathcal{J} \subseteq \mathcal{X}$. A term rewriting system $\mathcal{R}$ is $\mathcal{J}-$right-linear, if for every rule $l \to r$ of $\mathcal{R}$, all $p,q \in \mathcal{FP}\mathrm{os}(r)$, $r(p) = r(q)$ and $r(p) \in \mathcal{J}$ imply $p = q$.*

Now we can claim the following inclusion result.

Theorem 1. *Let $\mathcal{Z}_1,\ldots,\mathcal{Z}_k \subseteq \mathcal{X}$, $F_1,\ldots,F_k \subseteq \mathcal{F}_0$, and $f_1,\ldots,f_k \in \mathcal{F}_1$. Let $\mathcal{A} = (\mathcal{Q},\Delta,\mathcal{Q}_{\mathrm{final}})$ be a tree automaton, and γ be an approximation function such that $(\mathcal{A},\gamma)$ is a $(F_i,\mathcal{Z}_i,f_i)$-compatible pair for all $i \in \{1,\ldots,k\}$. If $\mathcal{R}$ is $(\mathcal{X}\setminus\bigcup_{i=1}^k(\mathcal{Z}_i))$-left-linear TRS then one has*

$$\mathcal{R}(\mathcal{L}(\mathcal{A})) \cup \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}((f_\gamma(\mathcal{A}),\gamma))$$

and $f_{\mathcal{R},\gamma}(\mathcal{A})$ is $(F_i,\mathcal{Z}_i,f_i)$-compatible for all $i \in \{1,\ldots,k\}$. Moreover, if $\mathcal{A}$ is a finite automaton then $f_{\mathcal{R},\gamma}(\mathcal{A})$ is finite too.

Proof. By construction the set Δ of transitions of $\mathcal{A}$ is included in the set of transitions of $(f_\gamma(\mathcal{A}), \gamma)$. It implies that

$$\mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(f_\gamma(\mathcal{A})).$$

Now we prove that $\mathcal{R}(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(f_\gamma(\mathcal{A}))$. Let $t \in \mathcal{L}(\mathcal{A})$, $l \to r \in \mathcal{R}$, $p \in \mathcal{P}\text{os}(t)$ and $\mu \in \Sigma(\mathcal{T}(\mathcal{F}), \mathcal{X})$ be a substitution such that

$$t|_p = l\mu. \tag{2}$$

We prove that $t[r\mu]_p \in \mathcal{L}(f_\gamma(\mathcal{A}))$. Since $t \in \mathcal{L}(\mathcal{A})$, there exists $q_f \in \mathcal{Q}_\text{f}$ such that

$$t \to^*_\Delta q_f. \tag{3}$$

Thus, by definition of a successful computation in a tree automaton, there exists $q \in \mathcal{Q}$ such that

$$t|_p \to^*_\Delta q \text{ and } t[q]_p \to^*_\Delta q_f. \tag{4}$$

Let $C[]$ be a ground context such that $l = C[x_1, \ldots, x_n]$ and $\{x_i \mid 1 \leq i \leq n\} = \mathcal{V}\text{ar}(l)$. One has

$$l\mu = C[\mu(x_1), \ldots, \mu(x_n)] \text{ and } C[\mu(x_1), \ldots, \mu(x_n)] \to^*_\Delta q. \tag{5}$$

Consequently, from (5) and using (2) and (4) one can deduce that there exist $q_1, \ldots, q_n \in \mathcal{Q}$ such that

$$C[q_1, \ldots, q_n] \to^*_\Delta q' \text{ and } \mu(x_1) \to^*_\Delta q_1, \ldots, \mu(x_n) \to^*_\Delta q_n. \tag{6}$$

We claim that there exists a substitution $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$ such that $\sigma(x_i) = q_i$ for all $i \in \{1, \ldots, n\}$. Let $j \in \{1, \ldots, n\}$. There are two cases:

- If the variable x_j occurs only once, i.e., if $\ell \neq j$ then $x_\ell \neq x_j$. In this case we set $\sigma(x_j) = q_j$.
- Otherwise, since $\mathcal{R}$ is $(\mathcal{X} \backslash \bigcup_{i=1}^k (\mathcal{Z}_i))$-left-linear, there exists $\ell \in \{1, \ldots, k\}$ such that $x_j \in \mathcal{Z}_\ell$. Consequently, by Lemma 2, $\mu(x_j) \in F_\ell$. Thus, by condition *1* of Definition 5, if $x_m = x_j$ then $q_j = q_m$. Consequently, we can set $\sigma(x_j) = q_j$.

Proving the claim.

Therefore and by (4), $l\sigma \to^*_\Delta q$. Consequently and by Lemma 1, $r\sigma \to^*_{f_\gamma(\mathcal{A})} q$. Moreover, using the claim above, one obtains $r\mu \to^*_{f_\gamma(\mathcal{A})} q$. By construction, Δ is included in the set of transitions of $f_{\mathcal{R},\gamma}$. Then, one has $t[r\mu]_p \to^*_{f_\gamma(\mathcal{A}))} q_f$ by (4), proving that $t[r\mu] \in \mathcal{L}((f_\gamma(\mathcal{A}), \gamma))$.

The $(F_i, \mathcal{Z}_i, f_i)$-compatibility of $(f_\gamma(\mathcal{A}), \gamma)$ is a direct consequence of Lemma 4.

Since $\mathcal{R}$ is finite, if Δ is finite then Δ' is finite too, and we are done.

Now, one can inductively apply the previous theorem. With the above notations, $f_\gamma(\mathcal{A})$ can be used to obtain an over-approximation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$. Formally,

Proposition 1. *If there exists a positive integer n such that $\mathcal{L}(f_\gamma^n(\mathcal{A})) = \mathcal{L}(f_\gamma^{n+1}(\mathcal{A}))$, then $\mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(f_\gamma^n(\mathcal{A}))$.*

Proof. We first claim that for all $N \geq 1$,

$$\mathcal{R}^N(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(f_\gamma^N(\mathcal{A})). \tag{7}$$

The proof is by induction.

- For $N = 1$, (7) is true by Theorem 1.
- Assume there exists N such that (7) is true. By Theorem 1, $f_\gamma^N(\mathcal{A})$ is a $(F_i, \mathcal{Z}_i, f_i)-$compatible automaton for all $i \in \{1, \ldots, k\}$. Thus, applying $\mathcal{R}$ to (7), one obtains

  $$\mathcal{R}(\mathcal{R}^N(\mathcal{L}(\mathcal{A}))) \subseteq \mathcal{R}(\mathcal{L}(f_\gamma^N(\mathcal{A}))). \tag{8}$$

  Next, Theorem 1 for $f_\gamma^N(\mathcal{A})$ implies

  $$\mathcal{R}(\mathcal{L}(f_\gamma^N(\mathcal{A}))) \subseteq \mathcal{L}(f_\gamma(f_\gamma^N(\mathcal{A}))). \tag{9}$$

  Now, using (8) and (9), one has

  $$\mathcal{R}^{N+1}(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(f_\gamma^{N+1}(\mathcal{A})),$$

  and we are done for the induction.

If there exists an integer $n \geq 1$ such that $\mathcal{L}(f_\gamma^n(\mathcal{A})) = \mathcal{L}(f_\gamma^{n+1}(\mathcal{A}))$, then one can deduce that for all $N \geq n$,

$$\mathcal{R}(\mathcal{L}(f_\gamma^N(\mathcal{A}))) \subseteq \mathcal{L}(f_\gamma^n(\mathcal{A})).$$

It implies that

$$\bigcup_{N \geq 0} \mathcal{R}^N(\mathcal{L}(\mathcal{A})) = \mathcal{R}^*(\mathcal{L}(\mathcal{A})) \subseteq \mathcal{L}(f_\gamma^n(\mathcal{A})).$$

5 Automatically Generated Approximations

In this section, we define two classes of approximation functions that can be automatically computed. The first one requires a right-linearity hypothesis (generally satisfied on security protocols) and provides a sequence of under-approximations of accessible terms. The idea is to use new states each time it is necessary (to do that, a function φ is introduced). Notice that this sequence converges to the exact set of accessible terms (Proposition 2), but not

necessary in a finite time. The second one provides an over-approximation of accessible terms in a finite time (Proposition 3). The function ψ is introduced for this approximation function. Practical applications on security protocols will be discussed in Section 6.

Let $\mathcal{Q}_0$ be a subset of $\mathcal{Q}$. Let $\mathcal{Z}_1, \ldots, \mathcal{Z}_k \subseteq \mathcal{X}$, $F_1, \ldots, F_k \subseteq \mathcal{F}_0$, $f_1, \ldots, f_k \in \mathcal{F}_1$ and $\mathcal{A} = (\mathcal{Q}_0, \Delta, \mathcal{Q}_f)$ be a tree automaton such that $\mathcal{A}$ is $(F_i, \mathcal{Z}_i, f_i) - \textit{compatible}$ for all $i \in \{1, \ldots, k\}$.

Let

$$\mathcal{Z} = \cup_{i=1}^{k} \mathcal{Z}_i \quad \textit{and} \quad F = \cup_{i=1}^{k} F_i \quad \textit{and} \quad \overline{\mathcal{Q}_0} = \cup_i Q_{F_i, \mathcal{Z}_i, f_i}(\mathcal{A}).$$

Let φ be an injective function from $\mathcal{R} \times \Sigma(\mathcal{Q}, \mathcal{X}) \times \mathcal{Q} \times \mathbb{N}^*$ into $\mathcal{Q} \setminus \mathcal{Q}_0$, F be a subset of $\mathcal{F}$, $k_0 \geq 0$, $\mathcal{Z}_0$ a subset of $\mathcal{Z}$ of cardinality k_0 and ψ be an injective function from $\mathcal{R} \times \mathbb{N}^* \times 2^{\mathcal{Q}}$ into $\mathcal{Q} \setminus \mathcal{Q}_0$ such that $\psi(\mathcal{R} \times \mathbb{N}^* \times 2^{\mathcal{Q}}) \cap \varphi(\mathcal{R} \times \Sigma(\mathcal{Q}, \mathcal{X}) \times \mathcal{Q} \times \mathbb{N}^*) = \emptyset$.

Let γ_φ be the approximation function defined by:

- $\gamma_\varphi(l \to r, \sigma, q)(y_{l \to r, p}) = \varphi(l \to r, \sigma, q, p)$ for all $l \to r \in \mathcal{R}$, $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$, $q \in \mathcal{Q}$, $p \in \mathcal{P}\text{os}(r)$ if $r(p) \notin F$,
- $\gamma_\varphi(l \to r, \sigma, q)(y_{l \to r, p}) = q_{r(p)}$ if $r(p) \in F$.

Notice that $(\mathcal{A}, \gamma_\varphi)$ is a $(F_i, \mathcal{Z}_i, f_i) - \textit{compatible}$ pair for all $i \in \{1, \ldots, k\}$.

The three following lemmas are useful to prove Proposition 2.

Lemma 5. *If $\mathcal{R}$ is $(\mathcal{X} \setminus \mathcal{Z})$-left and right linear then for all $t \in \mathcal{T}(\mathcal{F})$ and $\sigma \in \Sigma(\mathcal{Q}_0, \mathcal{X})$ such that $t \to_{\mathcal{A}}^* r\sigma \to_{f_{\gamma_\varphi}(\mathcal{A})}^* q$, $l\sigma \to_{\mathcal{A}}^* q$ and $q \in \mathcal{Q}_0$ There exists $t_0 \in \mathcal{T}(\mathcal{F})$ such that $t \in \mathcal{R}(t_0)$ and $t_0 \to_{\mathcal{A}}^* q$.*

Proof. Let $r = C[y_1, \ldots, y_n]$ and, for all $i \in \{1, \ldots, n\}$, p_i be the position of y_i in r. Since $t \to_{\mathcal{A}}^* r\sigma$, notice that all p_i's are also positions of t. We construct now a substitution $\mu \in \Sigma(\mathcal{T}(\mathcal{F}), \mathcal{X})$ as follows:

- If a variable y occurs only once in r, i.e., there exists only one i such that $y = y_i$, we set $\mu(y) = t_{|p_i}$.
- If a variable y occurs at least twice in r, we may assume (without lost of generality) that $y = y_1 = y_2$. We claim that $t_{|p_1} = t_{|p_2}$. Indeed, by the right-linearity condition, there exists $1 \leq i \leq k$ such that $y \in \mathcal{Z}_i$. Since $(\mathcal{A}, \gamma_\varphi)$ is $(F_i, \mathcal{Z}_i, f_i)$-compatible, there exist, by condition *6* of Definition 5, positions p'_1 and p'_2 of r such that $r_{|p'_1} = f_i(y)$, $r_{|p'_2} = f_i(y)$, $p'_1 1 = p_1$ and $p'_2 1 = p_2$. Moreover, by Lemma 4, $(f_{\gamma_\varphi}(\mathcal{A}), \gamma_\varphi)$ is $(F_i, \mathcal{Z}_i, f_i)$-compatible too. Consequently, using condition *1* of Definition 5, and since $r\sigma \to_{f_{\gamma_\varphi}(\mathcal{A})}^* q$, there exists $A \in F_i$ such that $t_{|p'_1} = f_i(A)$ and $t_{|p'_2} = f_i(A)$. We set $\mu(y) = A$.
- If a variable y does not occur in r, but occurs in l, we construct $\mu(y)$ as in the proof of Theorem 1.
- If a variable y does not occur neither in r, nor in l, we arbitrarily set $\mu(y)$ (this value will never be used).

By construction, one has $t = r\mu$ and for all variable y occurring in l, $\mu(y) \rightarrow_{\mathcal{A}}^{*} \sigma(y)$. Consequently $l\mu \rightarrow_{\mathcal{A}}^{*} l\sigma \rightarrow_{\mathcal{A}}^{*} q$. It implies the proof of the lemma with $t_0 = l\mu$.

Lemma 6. *Let $t \in \mathcal{T}(\mathcal{F})$, $q \in \mathcal{Q}_0$ such that $t \not\rightarrow_{\mathcal{A}}^{*} q$ and such that there exists a function τ from $\mathcal{P}\mathrm{os}(t)$ to Q satisfying the following conditions:*

(i) $\tau(\varepsilon) = q$ and,
(ii) For all $p \in \mathcal{P}\mathrm{os}(t)$, $t_{|_p} \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})} \tau(p)$ and
(iii) For all $p \in \mathcal{P}\mathrm{os}(t) \setminus \{\varepsilon\}$, if $\tau(p) \in \mathcal{Q}_0$, then $t_{|_p} \rightarrow_{\mathcal{A}}^{*} \tau(p)$.

Then there exists $t_0 \in \mathcal{T}(\mathcal{F})$ such that $t_0 \rightarrow_{\mathcal{A}}^{*} q$ and $t \in \mathcal{R}(t_0)$.

Proof. By (i) and (ii), there exists $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{Q})$ such that

$$t \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})}^{*} s_1 \rightarrow_{\gamma_\varphi} q.$$

The transition $s_1 \rightarrow q \notin \Delta$. Indeed, if $s_1 \rightarrow q \in \Delta$ then $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{Q}_0)$. Thus, by (iii), $t \rightarrow_{\mathcal{A}}^{*} s_1 \rightarrow_{\mathcal{A}} q$ which is a contradiction ($t \notin L(\mathcal{A}, q)$).

Consequently, there exist q', $\sigma \in \Sigma(\mathcal{X}, \mathcal{Q})$ and $l \rightarrow r \in \mathcal{R}$ such that $s_1 \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})} q \in \mathrm{Norm}_{\gamma_\varphi}(l \rightarrow r, \sigma, q')$ and

$$l\sigma \rightarrow_{\mathcal{A}}^{*} q'. \tag{10}$$

By definition of γ_φ and since $q \in \mathcal{Q}_0$, either $q = q'$ or $q \in \overline{\mathcal{Q}_0}$. Since $\mathcal{A}$ is $(F_i, \mathcal{Z}_i, f_i)$ – *compatible* for all $i \in \{1, \ldots, k\}$, if $q \in \overline{\mathcal{Q}_0}$, then $s_1 \rightarrow q \in \Delta$. It implies that $q = q'$ and that

$$t \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})}^{*} s_1 \rightarrow_{\mathrm{Norm}(l \rightarrow r, \sigma, q)} q.$$

If $s_1 \notin \mathcal{T}(\mathcal{F}, \mathcal{Q}_0)$ then there exists a position p of s_1 such that $s_1(p) \in \mathcal{Q} \setminus \mathcal{Q}_0$. Thus $s_1(p)$ is of the form $s_1(p) = \varphi(l \rightarrow r, \sigma, p, q)$. Since φ is injective, the only transition of $\rightarrow_{\gamma_\varphi}(\mathcal{A})$ leading to $s_1(p)$ is

$$r(p)(\gamma_\varphi(y_{l \rightarrow r, p.1}), \ldots, \gamma_\varphi(y_{l \rightarrow r, p.\ell})) \rightarrow s_1(p).$$

Let

$$s_2 = s_1[r(p)(\gamma_\varphi(y_{l \rightarrow r, p.1}), \ldots, \gamma_\varphi(y_{l \rightarrow r, p.\ell}))]_p.$$

One has

$$t \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})}^{*} s_2 \rightarrow_{\mathrm{Norm}(l \rightarrow r, \sigma, q)} s_1 \rightarrow_{\mathrm{Norm}(l \rightarrow r, \sigma, q)} q.$$

By induction, one can iterate this construction to build a term $s \in \mathcal{T}(F, \mathcal{Q}_0)$ such that

$$t \rightarrow_{f_{\gamma_\varphi}(\mathcal{A})}^{*} s \rightarrow_{\mathrm{Norm}(l \rightarrow r, \sigma, q)}^{*} q, \tag{11}$$

and for all position p of s such that $s(p) \notin \mathcal{Q}$,

$$s(p) = r(p). \tag{12}$$

Let $s = S[q_1, \ldots, q_n]$ and let p_ℓ be the position of q_ℓ in s. Let $s' = S[t_1, \ldots, t_n]$ with $t_\ell = q_\ell$ if $r(p_\ell) \in \mathcal{X}$ and $t_\ell = r(p_\ell)$ otherwise. By definition of $\mathrm{Norm}_{\gamma_\varphi}$ and using (11) and (12), one has

- If $q_\ell \notin \overline{\mathcal{Q}_0}$, then $r(p_\ell) \in \mathcal{X} \setminus \mathcal{Z}$ and $\sigma(r(p_\ell)) = q_\ell$,
- If $q_\ell \in \overline{Q}$, then either $r(p_\ell) \in \mathcal{Z}$ and $\sigma(r(p_\ell)) = q_\ell$, or $r(p_\ell) \in F$.

It follows that $s' = r\sigma$. Furthermore, using (11) and by construction, one has

$$t \rightarrow^*_{f_{\gamma_\varphi}(\mathcal{A})} r\sigma \rightarrow^*_{\mathrm{Norm}(l\rightarrow r,\sigma,q)} q.$$

Moreover, if $t_\ell \in \mathcal{Q}_0$ then $t_\ell = \tau(p_\ell)$. Thus, by (iii), $t_{|p_\ell} \rightarrow^*_{\mathcal{A}} t_\ell$. Consequently, we obtain

$$t \rightarrow^*_{\mathcal{A}} r\sigma \rightarrow^*_{\mathrm{Norm}(l\rightarrow r,\sigma,q)} q.$$

Therefore, using (10) and Lemma 5, there exists $t_0 \in \mathcal{T}(\mathcal{F})$ such that $t_0 \rightarrow^*_{\mathcal{A}} q$ and $t \in \mathcal{R}(t_0)$, proving the lemma.

Lemma 7. *If $\mathcal{R}$ is $(\mathcal{X} \setminus \mathcal{Z})$-left and right-linear then one has*

$$L(f_{\gamma_\varphi}(\mathcal{A})) \subseteq \mathcal{R}^*(L(\mathcal{A})).$$

Proof. Let $\mathcal{P}_n$ be the following proposition:
For all $t \in \mathcal{L}(f_{\gamma_\varphi}(\mathcal{A}))$, if there exists a function τ from $\mathcal{P}\mathrm{os}(t)$ to $\mathcal{Q}$ such that $\tau(\varepsilon) = q_f$ and for all $p \in \mathcal{P}\mathrm{os}(t)$,

$$t|_p \rightarrow^*_{f_{\gamma_\varphi}(\mathcal{A})} \tau(p) \quad \textit{and} \quad t[\tau(p)]_p \rightarrow^*_{f_{\gamma_\varphi}(\mathcal{A})} q_f$$

and such that

$$|\{p \in \mathcal{P}\mathrm{os}(t) \mid \tau(p) \in \mathcal{Q}_0 \ \wedge \ t|_p \not\rightarrow^*_{\mathcal{A}} \tau(p)\}| = n,$$

then $t \in \mathcal{R}^*(\mathcal{L}(\mathcal{A}))$.
We prove that $\mathcal{P}_n$ is true for all $n \geq 0$ by induction on n. To simplify notations, let

$$NR(t,\tau) = \{p \in \mathcal{P}\mathrm{os}(t) \mid \tau(p) \in \mathcal{Q}_0 \textit{ and } t|_p \not\rightarrow^*_{\mathcal{A}} \tau(p)\}.$$

$\mathcal{P}_0$: Assume that t and τ follow the hypothesis of $\mathcal{P}_0$. We have $|NR(t,\tau)| = 0$. In particular, $\varepsilon \notin NR(t,\tau)$ so $t = t|_\varepsilon \rightarrow_{\mathcal{A}} \tau(\varepsilon) = q_f$. Since $\mathcal{A}$ and $f_{\gamma_\varphi}(\mathcal{A})$ have the same set of final states, $t \in \mathcal{L}(\mathcal{A})$.

$\mathcal{P}_n \implies \mathcal{P}_{n+1}$: Assume that $\mathcal{P}_n$ is true for $n \geq 0$ and that t and τ follow the hypothesis of $\mathcal{P}_{n+1}$. Since $NR(t,\tau)$ is non-empty, let p be a maximal element of $NR(t,\tau)$ (for the lexicographical order). By maximality of p, one can apply Lemma 6 to $t_{|p}$. Thus, there exists $t_0 \in \mathcal{T}(\mathcal{F})$ such that $t_0 \rightarrow^*_{\mathcal{A}} \tau(p)$ and $t_p \in \mathcal{R}(t_0)$. Therefore, there exists a function τ_1 from $\mathcal{P}\mathrm{os}(t_0)$ in $\mathcal{Q}_0$ such that for all p', $t_0 \rightarrow^*_{\mathcal{A}} \tau_1(p')$, $t[\tau_1(p')]_{p'} \rightarrow^*_{f_{\gamma_\varphi}(\mathcal{A})} \tau(p)$. We define the function τ_2 from $\mathcal{P}\mathrm{os}(t[t_0]_p)$ to $\mathcal{Q}$ by

- If p is not a prefix of p', then $\tau_2(p') = \tau(p')$,
- If p' is of the form pu, then $\tau_2(p') = \tau_1(u)$.

By construction, $t \in \mathcal{R}(t[t_0]_p)$ and $|NR(t[t_0]_p, \tau_2)| = n - 1$. Thus, by induction, $t \in \mathcal{R}^*(\mathcal{L}(\mathcal{A})))$.

Let $\mathcal{B}_n(\mathcal{A})$ be the tree automaton defined by $\mathcal{B}_n(\mathcal{A}) = f^n_{\gamma_\varphi}(\mathcal{A})$. In the next proposition, we prove that the approximation function γ_φ provides an under-approximation of accessible terms.

Proposition 2. *If $\mathcal{R}$ is $(\mathcal{X} \setminus \mathcal{Z})$-left and right-linear then for all $n \leq 0$, $L(\mathcal{B}_n(\mathcal{A})) \subseteq \mathcal{R}^*(L(\mathcal{A}))$, $L(\mathcal{B}_n(\mathcal{A})) \subseteq L(\mathcal{B}_{n+1}(\mathcal{A}))$ and*

$$\bigcup_{n \geq 0} L(\mathcal{B}_n(\mathcal{A})) = \mathcal{R}^*(L(\mathcal{A})).$$

Proof. By definition $\mathcal{B}_{n+1}(\mathcal{A}) = f_\gamma(\mathcal{B}_n(\mathcal{A})))$. Consequently, the set of transitions of $\mathcal{B}_n(\mathcal{A})$ is included in the transitions set of $\mathcal{B}_{n+1}(\mathcal{A})$. Thus $L(\mathcal{B}_n(\mathcal{A})) \subseteq L(\mathcal{B}_{n+1}(\mathcal{A}))$.

Now, by Theorem 1, one has for all $n \geq 1$,

$$\mathcal{R}(L(\mathcal{B}_n(\mathcal{A}))) \cup L(\mathcal{B}_n(\mathcal{A})) \subseteq L(\mathcal{B}_{n+1}(\mathcal{A}).$$

Consequently, by a direct induction,

$$\mathcal{R}^{\leq n}(\mathcal{A}) \subseteq L(\mathcal{B}_{n+1}(\mathcal{A}).$$

It implies that

$$\mathcal{R}^*(L(\mathcal{A})) \subseteq \bigcup_{n \geq 0} L(\mathcal{B}_n(\mathcal{A})).$$

One can prove that for all $n \in \mathbb{N}$, $L(\mathcal{B}_n(\mathcal{A})) \subseteq \mathcal{R}^*(L(\mathcal{A}))$ by a direct induction on n using Lemma 7.

We now address the question of an over-approximation. Let $z_1, \ldots, z_{k_0}$ be the elements $\mathcal{Z}$. Let $\mathcal{C}(\mathcal{A})$ be the tree automata defined by $\mathcal{C}(\mathcal{A}) = f_{\gamma_\mathcal{A}}(\mathcal{A})$ where $\gamma_\mathcal{A}$ is inductively defined for all $l \to r \in \mathcal{R}$, $\sigma \in \Sigma(\mathcal{Q}, \mathcal{X})$, $q \in \mathcal{Q}$, $p \in \mathcal{P}\text{os}(r)$ by

- if $r(p) \in \mathcal{F}_0$, then $\gamma_\mathcal{A}(l \to r, \sigma, q)(y_{l \to r,p})$ is equal to

$$\min\{q \mid r(p) \to q \in \Delta\}$$

if it exists, and to
$\psi(l \to r, p, \{\sigma(z) \mid z \in \mathcal{Z} \cap \mathcal{V}\text{ar}(l))$ otherwise.

– if $r(p) \in \mathcal{F}_\ell$ $(\ell > 0)$, then $\gamma_{\mathcal{A}}(l \to r, \sigma, q)(y_{l \to r,p})$ is equal to

$$\min\{q \mid r(p)(\beta_1, \ldots, \beta_n) \to q \in \Delta) \mid \begin{array}{l}\beta_i = r(p.i)\sigma \text{ if } r(p.i) \in \mathcal{X}, \\ \beta_i = \gamma_{\mathcal{A}}(l \to r, \sigma, q)(y_{l \to r,p.i}) \\ \text{otherwise } \}\end{array}$$

if it exists and to
$\psi(l \to r, p, \{\sigma(z) \mid z \in \mathcal{Z} \cap \mathcal{V}\mathrm{ar}(l))$ otherwise.

For all $n, i \leq 0$, $\mathcal{C}_n^i(\mathcal{A})$ is inductively defined by $\mathcal{C}_n^0(\mathcal{A}) = \mathcal{C}(\mathcal{B}_n(\mathcal{A}))$ and $\mathcal{C}_n^{i+1}(\mathcal{A}) = \mathcal{C}(\mathcal{C}_n^i(\mathcal{A}))$. In the next proposition, we prove that the approximation function γ_ψ provides an over-approximation of accessible terms.

Proposition 3. *If $\mathcal{R}$ is $(\mathcal{X} \setminus \mathcal{Z})$-left-linear and if $\mathcal{A}$ is finite then the sequence $(\mathcal{C}_n^k(\mathcal{A}))_{k \leq 0}$ is ultimately constant for all $n \geq 0$. We denote by $\mathcal{C}_n(\mathcal{A})$ its limit. The tree automaton $\mathcal{C}_n(\mathcal{A})$ is finite. Furthermore, for all $n \leq 0$, $\mathcal{R}^*(L(\mathcal{A})) \subseteq L(\mathcal{C}_n(\mathcal{A}))$.*

Proof. Since $(\mathcal{A}, \gamma_\varphi)$ is a $(F_i, \mathcal{Z}_i, f_i) - compatible$ pair for all $i \in \{1, \ldots, k\}$, by induction and using Lemma 4, $(\mathcal{B}_n(\mathcal{A}), \gamma_\varphi)$ is also a $(F_i, \mathcal{Z}_i, f_i) - compatible$ pair and

$$\overline{\mathcal{Q}_0} = \cup_i Q_{F_i,\mathcal{Z}_i,f_i}(\mathcal{B}_n(\mathcal{A})) = \cup_i Q_{F_i,\mathcal{Z}_i,f_i}(\mathcal{A}) = \cup_i Q_{F_i,\mathcal{Z}_i,f_i}(\mathcal{C}_n^k(\mathcal{A})).$$

Let $q \in \mathcal{Q}$, $l \to r$ be a rule of $\mathcal{R}$ and σ a substitution of $\Sigma(\mathcal{X}, \mathcal{Q})$ such that

$$l\sigma \to^*_{\mathcal{B}_n(\mathcal{A})} q.$$

Since $(\mathcal{C}_n^k(\mathcal{A}), \gamma_\varphi)$ is $(F_i, \mathcal{Z}_i, f_i) - compatible$, if $z \in \mathcal{Z} \cap \mathcal{V}\mathrm{ar}(l)$, then

$$\sigma(z) = q_{\sigma(z)} \in \overline{\mathcal{Q}_0}.$$

Consequently, $\{\sigma(z) \mid z \in \mathcal{Z} \cap \mathcal{V}\mathrm{ar}(l)) \subseteq 2^{\overline{\mathcal{Q}_0}}$ is finite (since $\overline{\mathcal{Q}_0}$ is finite). Therefore, the set of states of $\mathcal{C}_n^k(\mathcal{A})$ is included in $\mathcal{Q}(\mathcal{B}_n(\mathcal{A})) \cup \psi(\mathcal{R} \times \{\mathcal{P}\mathrm{os}(r) \mid l \to r \in \mathcal{R}\} \times 2^{\overline{\mathcal{Q}_0}}$ which is a finite set. Moreover, for all n, k, the set of states of $\mathcal{C}_n^k(\mathcal{A})$ is included in the set of states of $\mathcal{C}_n^{k+1}(\mathcal{A})$. The same arguments hold for transitions. Consequently, the sequence $(\mathcal{C}_n^k(\mathcal{A}))_{k \leq 0}$ is ultimately constant for all $n \geq 0$.

The inclusion $\mathcal{R}^*(L(\mathcal{A})) \subseteq L(\mathcal{C}_n(\mathcal{A}))$ is a direct consequence of Theorem 1.

Propositions 2 and 3 provide several upper and under-approximations of the set of accessible terms as shown in Fig. 2. Using notations of Section 2.1, one can use defined approximations in the semi-algorithm presented in Fig. 3.

Basically, when this semi-algorithm terminates, either the condition $A = \emptyset$ or the condition $E \neq \emptyset$ is be false. If $A \neq \emptyset$ then at least one secret term is reached (the property is violated) and if $E = \emptyset$ then there are not any secret terms in the over-approximation computed (the property is safe).

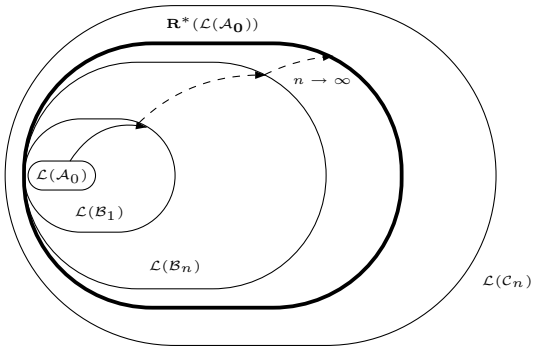

Fig. 2. Inclusions of languages involved in Propositions 2 and 3

$A := \mathcal{L}(\mathcal{B}_0(\mathcal{A})) \cap \mathcal{L}(\mathcal{A}_{\text{secret}})$;
$E := \mathcal{L}(\mathcal{C}_0(\mathcal{A})) \cap \mathcal{L}(\mathcal{A}_{\text{secret}})$;
$n := 0$;
while ($A = \emptyset$ **and** $E \neq \emptyset$) **do**
    $n := n + 1$;
    $A := \mathcal{L}(\mathcal{B}_n(\mathcal{A})) \cap \mathcal{L}(\mathcal{A}_{\text{secret}})$;
    $E := \mathcal{L}(\mathcal{C}_n(\mathcal{A})) \cap \mathcal{L}(\mathcal{A}_{\text{secret}})$;
endwhile
if ($A \neq \emptyset$)
    then return false;
    else return true;
endif

Fig. 3. Semi-algorithm using automatically generated approximation

6 Application to Security Protocols

This section is dedicated to the application of obtained results, introduced previously, to the practical automatic verification of security protocols. In Section 6.1, we introduce the high level specification language (`HLPSLin` [4]) used to specify protocols and we explain how it is translated into the formalism of tree automata and TRS. Then, in Section 6.2 we describe the `TA4SP`[4] tool which provides practical results on well-known protocols presented in Subsection 6.3 and on industrial security protocols in Section 6.4. Notice that only two agents are used to check secrecy property following results in [8].

[4] Tree Automata based Automatic Approximations for the Analysis of Security Protocols developed in LIFC.

6.1 Specifying Protocols with Term Rewriting Systems and Tree Automata

Using TRS and/or tree automata to encode security protocols is common, see for example [12,18,15]. In this section, we describe the specification language used by our tool and we explain how it is translated to a TRS.

We use the protocol specification language HLPSL (High Level Protocol Specification Language) defined in [4] and briefly presented in this section. Notice that HLPSL is used by other tools [1,2,19].

Specification Language Protocol specifications in HLPSL are divided into *roles*. Some roles (the so-called *basic* roles) describe the actions of one single agent in a run of a protocol or sub-protocol. Others (*composed* roles) instantiate these basic roles to model an entire protocol run (potentially consisting of the execution of multiple sub-protocols), a session of the protocol between multiple agents, or the protocol model itself. This latter role is often called the Environment role. For instance, the role bellow is Alice's role in the well-known Needham-Schroeder Public Key Protocol corrected version.

```
role Alice (A,B : agent,
            Ka,Kb : public_key,
            Snd,Rcv : channel (dy)) played_by A def=
  local
State : nat,
Na : text (fresh),
Nb : text
  init
State = 0
  knowledge(A) = A,B,Ka,Kb,inv(Ka)
  transition
    1. State=0 ∧ Rcv(start) =|>
          Snd({A.Na'}_Kb)
       ∧ State'=1
    2. State=1 ∧ Rcv({Na.Nb'.B}_Ka) =|>
          Snd({Nb'}_Kb)
       ∧ State'=2
       ∧ secret(Nb',A)
       ∧ secret(Nb',B)
end role
```

This role describes what Alice does in a NSPK protocol session. The first transition is the start of the protocol where Alice sends a nonce Na and her name A to Bob. Notice that a primed variable means that this variable is initialized. Thus, when this variable occurs in the right-hand side and not in the left-hand side of a rule, it is a fresh element. Thus, the second transition means that:

- Alice receives a concatenation of 3 elements Na, Nb' and B where Na is the nonce created in transition 1, Nb' is the nonce generated by Bob and B is the Bob's identity,

– Alice sends Nb' encoded by Kb.

Then, given a set of roles describing the protocol and an `Environment` role, we define our security goals. In `Environment` role, we define the concrete sessions whose execution we want to consider. Currently, the specification language `HLPSL` supports only authentication and secrecy goals.

An `HLPSL` specification is translated into an IF file (Intermediate Format) used by several tools such as SATMC in [1], OFMC in [2], ATSE in [19] and `TA4SP` to check the properties specified.

The `TA4SP` tool needs to generate a TRS, an approximation function and tree automata: an initial automaton $\mathcal{A}_0$ and secret automaton $\mathcal{A}_{secret}$ which recognizes all terms supposed to be secret.

Term Rewriting System The TRS $\mathcal{R}$ represents the protocol steps and the intruder abilities to analyze a message.

For instance, the rule below corresponds to the second transition of the example above.

```
state_Alice(ag(A),ag(B),pk(Ka),pk(Kb),n1(ag(A),ag(B)),D_Nb,
            msg(crypt(pk(Ka),pair(n1(ag(A),ag(B)),
                                  n2(ag(X),ag(Y))))))
→
state_Alice(ag(A),ag(B),pk(Ka),pk(Kb),n1(ag(A),ag(B)),
            n2(ag(X),ag(Y)),
            msg(crypt(pk(Kb),n2(ag(X),ag(Y)))))
```

Notice that functional symbols are denoted by words beginning with a lower-case letter, and variables are denoted by words beginning with a capital letter. The symbols ag, pk, $pair$ respectively represent an agent, a public key, a constant and a concatenation operator. The fresh elements are expressed by the symbols $n1$ and $n2$.

The TRS also expresses certain abilities of the intruder. Indeed, by using the transitions below, the intruder can decrypt a message and analyze each component of a pair.

```
pair(X, Y) → Y
pair(X, Y) → X
pair(inv(pky(X)), crypt(pk(X), Y)) → Y
pair(pk(X), crypt(inv(pk(X)), Y)) → Y
```

Tree automata Two tree automata $\mathcal{A}_0$ and $\mathcal{A}_{secret}$ are generated from an IF specification. The tree automaton $\mathcal{A}_0$ encodes initial intruder knowledge and abilities of the intruder for the messages composition, while $\mathcal{A}_{secret}$ encodes all secret terms.

The composition ability of the intruder is handled by pertinent transitions in $\mathcal{A}_0$. The state q_f is a final state of $\mathcal{A}_0$, such that any term $t \in \mathcal{T}(\mathcal{F})$ which can be reduced to q_f is in the initial intruder knowledge. Then, using the transition $pair(q_f, q_f) \rightarrow q_f$, the intruder can compose as many messages with unbounded size as he wishes. Obviously, he

can also encode messages using a similar rule (modulo the functional symbol). Moreover, $\mathcal{A}_0$ recognizes the following terms: $ag(a)$, $ag(i)$, $pk(k_a)$, $pk(k_i)$ and $inv(pk(k_i))$. Notice that each constant $a,\ i,\ k_a$ and k_i is associated to a unique state of $\mathcal{A}_0$.

Thus, setting $\mathcal{Z}_1 = \{A, B, X, Y\}$, $\mathcal{Z}_2 = \{Ka, Kb\}$, $F_1 = \{a, i\}$, $F_2 = \{ka, kb\}$, $f_1 = \{ag\}$ and $f_2 = \{pk\}$, $\mathcal{R}$ and $\mathcal{A}_0$ follow the conditions of Definition 5. For the TRS $\mathcal{R}$, constants or variables (substitutable by a constant) are protected by the same functional symbol in left-hand side as in the right-hand side of each rule of $\mathcal{R}$. Furthermore, these variables are the only ones allowed to appear at least twice in one side or in the other one of a rule. Thus, this TRS is J-left-linear and J-right-linear where $J = X \setminus \mathcal{Z}$ and $\mathcal{Z} = \{A, B, X0, X1, Ka, Kb\}$. For the automaton $\mathcal{A}_0$, each constant is protected by a functional symbol, and moreover each constant is associated to a unique state of $\mathcal{A}_0$.

The tree automaton $\mathcal{A}_{secret}$ has to be generated too. This tree automaton recognizes all terms supposed to be secret. For instance, in the example presented above, the nonce Nb has been translated into $n2(ag(X), agt(Y))$. Thus, the term $secret(Nb', A)$, in the `HLPSL` specification described in Section 6.1, remains to be added in $\mathcal{A}_{secret}$ by the following transitions:

$$a \rightarrow q_1,\ ag(q_1) \rightarrow q_2,\ and\ n2(q_2, q_2) \rightarrow q_{secret}$$

where q_{secret} is a final state of $\mathcal{A}_{secret}$.

The last important point concerns the approximation function $\gamma_{\mathcal{A}}$ dealing with the approximated case.

Approximation Function Since the number of agents is restricted to two following [8], F_i, f_i and $\mathcal{Z}_i$ are fixed. Thus, following the description presented at the end of Section 5, one can easily define the approximation function $\gamma_{\mathcal{A}}$.

Following this description, $\gamma_{\mathcal{A}}$ satisfies the conditions of Definition 5, since $\mathcal{R}$ and $\mathcal{A}_0$ follow this definition too and since for all the constants in F_i where $\gamma_{\mathcal{A}}$ has to normalise them, $\gamma_{\mathcal{A}}$ associates the unique possible state to these constants. Thus, $f_{\gamma_{\mathcal{A}}}$ has the properties given by Lemma 4, Theorem 1 and Proposition 1. The protocol verification can be done.

6.2 Tree Automata based Automatic Approximations for the Analysis of Security Protocols

The symbolic approximation approach has been implemented (the architecture of the tool, called `TA4SP`, is displayed in Fig. 4).

It combines the existing Timbuk2[5] library, and the following tools we have been implementing.

- A Timbuk2 package which computes a tree automaton recognising an over-approximation of the reachable terms as defined in Section 2.1. As for Timbuk2, its implementation is done in `OCAML`.

[5] An `OCAML` tree automata library developed by T. Genet at IRISA-Rennes under GPL. Notice that we use the second version of Timbuk (Timbuk2).

- A translator from the IF into Timbuk2 input file, called *IF2Timbuk2*. This tool generates:
  - the TRS from the `section rules` of the IF specification,
  - the initial automaton from the `section inits` of the IF specification,
  - the set of undesirable configurations (nonces, keys, etc.) as a tree automaton $\mathcal{A}_{secret}$ from the `section goals` of the IF specification, and
  - the approximation function used for the normalisation from the `section rules` of the IF specification.

Timbuk2 is a collection of tools for achieving proofs of reachability over TRS and for manipulating tree automata (bottom-up non-deterministic finite tree automata). The `TA4SP` input is an IF specification which is generated from an `HLPSL` specification by HLPSL2IF. Then *IF2Timbuk2* generates a specification for Timbuk2 which computes an over or under approximation of the reachable terms. Thus, the intersection between this automaton and $\mathcal{A}_{secret}$ is computed, which leads to generate an output containing:

- a description of the method used (over-approximation or not);
- the result obtained by `TA4SP` which is "true", "false" or "don't know". The answer "true" means that the protocol is secure (empty intersection with secret terms), while the answer "false" means that there is a potential attack (however, this cannot be certain since we have used approximations and abstractions). The tools answers "don't know" when none of the above answers can be provided.

6.3 Protocols NSPK, NSPK-Lowe, RSA

These protocols NSPK, NSPK-Lowe and RSA have been checked in the under-approximation context by `TA4SP` and well-known results have been confirmed.

For NSPK and NSPK-Lowe, the completion program terminates since the TRS associated allows to do it. Thus, $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ is exactly computed.

Concerning NSPK protocol, the secrecy of a fresh element is not verified. Therefore, this same property has been verified on the corrected version NSPK-Lowe protocol, and there is no attack in our model.

Concerning RSA protocol, the context of a run is that each agent does not know the public key of the others. Thus knowing x and thinking to communicate with the agent w, the agent w cannot verify the validity of a message `crypt(pk(y),x)` because w does not know `pk(y)`. The computation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ does not terminate for this protocol, but by computing $\mathcal{B}_3(\mathcal{A}_0) \cap \mathcal{A}_{secret}$, the secrecy property is not verified.

In order to make the computation stop, the approximation is used for $n \geq 3$. Thus, the secrecy property is not safe too.

Using the under-approximation context for these three protocols, `TA4SP` gives a positive result[6] for each of them.

[6] A positive result is true or false. True and false mean respectively that there is no attack and there is an attack in our verification context.

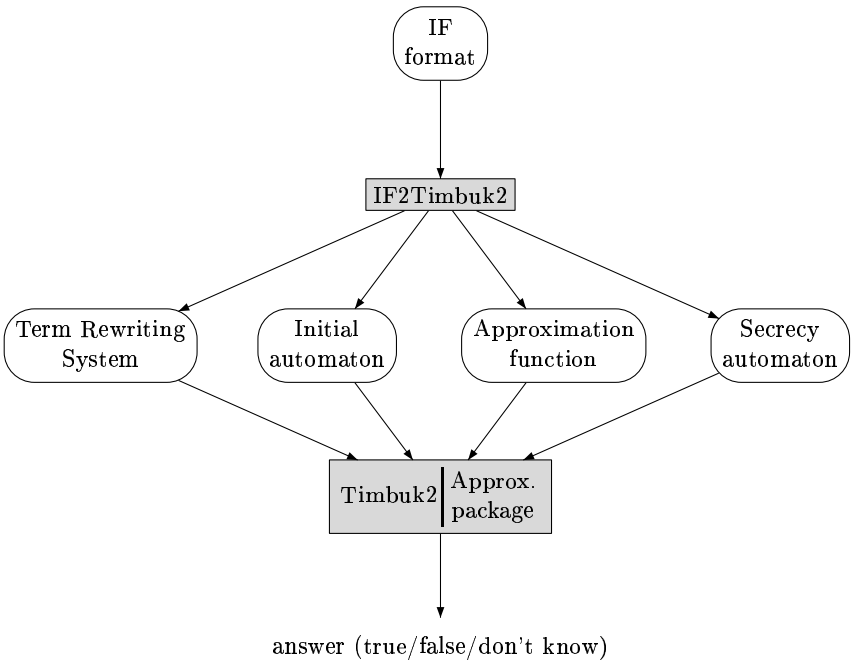

Fig. 4. TA4SP

6.4 Other Practical Results

For most of protocols listed in Table 1, the computation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ is not possible. Moreover, there is no positive result for any $n > 1$. Thus, an over approximation of $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$ is computed for each of them.

Table 1. Practical Results

Protocol	*Secrecy*	*Time (s)*
CHAPv2	verified	10.59
EKE	verified	49.24
SHARE	verified	48.97
IKEv2-CHILD	verified	274.92
TLS	verified	157.28
EAP_SIM	verified	2146.45

For all tested protocols (see Table 1), the intersection between the over-approximation of the reachable terms and the set of secret terms is empty; all the protocols are verified. These results show that the abstractions and the approximations used are not too coarse. Notice that for the protocols *IKEv2-CHILD* and *SHARE*, we have modified the TRS by renaming some non left-linear variables. The experiments show that the computation time depends on the size of the TRS, on the number of variables occurring in the rules and on the renaming of non left-linear variables. This latter case is an important point. Indeed, in a real execution of the protocol, an agent can compare an element to another one. Our approach does not allow this. And thus, a large number of acceptable[7] messages could be composed by the intruder. Among all these messages, many of them would not have to be accepted by the agents in a real execution of the protocol.

The tests have been performed on a Pentium4 2.4GHz with 640 MB of RAM.

7 Conclusion

In this article, we have presented a fully automatic approach to verify security protocols with an unbounded number of sessions. Practical tests on industrial security protocols, related to Internet communications, show the efficiency of this approach. Currently we only verify secrecy properties. We intend to develop techniques to automatically verify other properties, such as authentication. Recent work [16] shows that using associative-commutative finite tree automata is relevant to encode security protocols. We are interested in prospecting whether our automatically generated approximation functions may be used in the framework of [16].

References

1. A. Armando, L. Compagna, and P. Ganty. SAT-based Model-Checking of Security Protocols using Planning Graph Analysis. In K. Araki, S. Gnesi, and D. Mandrioli, editors, *Proceedings of the 12th International Symposium of Formal Methods Europe (FME)*, LNCS 2805, pages 875–893. Springer-Verlag, 2003. Available at *www.avispa-project.org*.
2. *D. Basin, S. Mödersheim, and L. Viganò. An On-The-Fly Model-Checker for Security Protocol Analysis. In Einar Snekkenes and Dieter Gollmann, editors,* Proceedings of ESORICS'03, *LNCS 2808, pages 253–270. Springer-Verlag, 2003. Available at http://www.avispa-project.org.*
3. *D. Bolignano. An approach to the formal verification of cryptographic protocols. In* ACM Conference on Computer and Communications Security, *pages 106–118, 1996.*
4. *Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drielsma, J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In* Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS), *Linz, Austria, September 2004. (13 pages).*

[7] For a given rule of the TRS, the pattern matching between the left hand side and an existing term represents the fact that the agent receives an expected–well-formed– message and can compose the next one.

5. *Y. Chevalier and L. Vigneron. Automated Unbounded Verification of Security Protocols. In Ed Brinksma and Kim Guldstrand Larsen, editors,* Proceedings of CAV'02, *LNCS 2404, pages 324–337. Springer-Verlag, 2002.*
6. *E. M. Clarke, S. Jha, and W. Marrero. A machine checkable logic of knowledge for specifying security properties of electronic commerce protocols. In* Workshop on Formal Methods and Security Protocols, *1998.*
7. *H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications, 2002. http://www.grappa.univ-lille3.fr/tata/.*
8. *H. Comon-Lundh and V. Cortier. Security properties: two agents are sufficient. In* Proceedings of ESOP'2003, *LNCS 2618, pages 99–113. Springer-Verlag, 2003.*
9. *P. Cousot. Abstract interpretation.* ACM Computing Surveys, *28(2):324–328, June 1996.*
10. *A. Durante, R. Focardi, and R. Gorrieri. CVS: A compiler for the analysis of cryptographic protocols. In* Proceedings of the 12th IEEE Computer Security Foundations Workshop (CSFW '99), *pages 203–213, Washington - Brussels - Tokyo, June 1999. IEEE.*
11. *G. Feuillade, T. Genet, and V. VietTriemTong. Reachability analysis over term rewriting systems.* Journal of Automated Reasonning, *2004. To appear.*
12. *Th. Genet and F. Klay. Rewriting for cryptographic protocol verification. In* Proceedings of CADE'00, *LNCS 1831, pages 271–290. Springer-Verlag, 2000.*
13. *G. Lowe. Some new attacks upon security protocols. In IEEE Computer Society Press, editor,* In Proceedings of the Computer Security Foundations Workshop VIII, *1996.*
14. *J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murϕ. In* Proceedings of the 1997 Conference on Security and Privacy (S & P-97), *pages 141–153, Los Alamitos, May 4–7 1997. IEEE Press.*
15. *D. Monniaux. Abstracting cryptographic protocols with tree automata. In* Sixth International Static Analysis Symposium (SAS'99), *number 1694 in Lecture Notes in Computer Science. Springer-Verlag, 1999.*
16. *H. Ohsaki and T. Takai. Actas: A system design for associative and commutative tree automata theory. In* Proceedings of the 5th International Workshop on Rule-Based Programming: RULE'2004, *Aachen, Germany, June 2004. To appear in ENTCS,.*
17. *Lawrence C. Paulson. Mechanized proofs for a recursive authentication protocol. In* 10th Computer Security Foundations Workshop, *pages 84–95. IEEE Computer Society Press, 1997.*
18. *M. Rusinowitch and M. Turuani. Protocol insecurity with finite number of sessions is NP-complete. In* 14th IEEE Computer Security Foundations Workshop (CSFW '01), *pages 174–190, Washington - Brussels - Tokyo, June 2001. IEEE.*
19. *J. Santos Santiago. Analyse automatique de protocoles avec AtSe. In J. Julliand, editor,* Congrès Approches Formelles dans l'Assistance au Développement de Logiciels, AFADL'04, *Besançon, France, June 2004. Présentation système.*
20. *S. Schneider. Verifying authentication protocols with CSP. In* PCSFW: Proceedings of The 10th Computer Security Foundations Workshop. *IEEE Computer Society Press, 1997.*

Unité de recherche INRIA Lorraine
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399